Summer of Code

BlockChain

Anuj Yadav

IIT BOMBAY

June 2025

Contents

CHAPTER 1

Introduction

This is a midterm Report for my summer of code project, and hence in this sense is still incomplete, and will be completed by the time of final submission.

CHAPTER 2

What is BlockChain ?

Blockchain Technology is the name given to a modern database management technique, which enables a large business to share information and communicate with the help of links and chains between different nodes in the network.

Blockchain serves as a distributed ledger, securely storing data across a network of computers. This sets it apart from centralized systems, where information is stored in a single location controlled by a central authority. Operating on a decentralized network, blockchain enables every participant to have a copy of the entire ledger, eliminating the need for intermediaries and fostering digital trust among users.

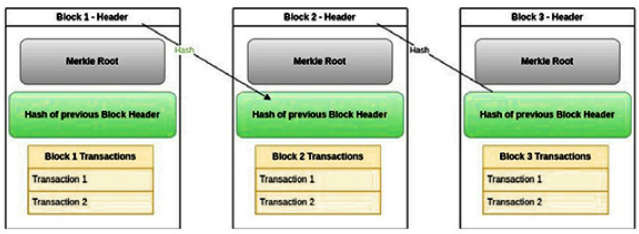

Figure 2.1: Simplified View of Blockchain Networks

2.1 Cryptography in Blockchain

Some important less known terms in cryptography:-

- **Digital Signature**
  A digital signature is a cryptographic code generated using the sender's private key and attached to an electronic message or document. It serves two primary purposes:
  - *Integrity*: Ensures the contents have not been altered in transit.
  - *Authentication*: Confirms the identity of the sender.

  Digital signatures are based on asymmetric cryptography, where the recipient can use the sender's public key to verify the signature.

- **Public Key Infrastructure (PKI)**
  PKI is a comprehensive system that manages digital keys and certificates. It includes:
  - *Hardware and software* for key generation and storage.
  - *Protocols* for key exchange, certificate issuance, and revocation.

  - *People and policies* governing trust relationships.

  PKI enables secure electronic transactions and communications by ensuring the authenticity of public keys via digital certificates.

- **Certificate Authority (CA)**
  A CA is a trusted entity responsible for:

  - Issuing digital certificates that bind a public key to an entity's identity.
  - Verifying identities before issuing certificates.
  - Managing the lifecycle of certificates (renewal, suspension, revocation).

  The CA plays a central role in PKI and helps establish a chain of trust.

- **Non-repudiation**
  Non-repudiation refers to the assurance that someone cannot deny the authenticity of their signature or the sending of a message. It involves:

  - *Proof of origin*: Linking actions or messages to specific individuals.
  - *Accountability*: Preventing denial of previous commitments or transmissions.

  Digital signatures and secure logging mechanisms are common tools to ensure non-repudiation.

Cryptography is the technique of securing information by converting it into a code that can only be read by those who are permitted to do so. The core principles and concepts of cryptography include:

- **Confidentiality**
  Ensures that sensitive data is accessible only to those authorized to view it. This prevents unauthorized access or disclosure of information during transmission or storage.

- **Integrity**
  Guarantees that the data has not been altered or tampered with in any unauthorized way. Even small changes to data should be detectable.

- **Authentication**
  Verifies the identity of the entities (sender and receiver) involved in communication. This ensures that the data originates from a trusted source.

- **Non-repudiation**
  Ensures that a sender cannot deny having sent a particular message. This is achieved through cryptographic proofs such as digital signatures.

- **Key Management**
  Involves the secure generation, distribution, storage, rotation, and destruction of cryptographic keys used in encryption and decryption processes.

- **Algorithms**
  Cryptographic algorithms are mathematical procedures that transform plaintext into ciphertext and vice versa. These algorithms form the backbone of secure communication.

- **Public Key Cryptography**
  A cryptographic method that uses a pair of keys:

  - A *public key* that is shared openly and used to encrypt data.

  - A *private key* that is kept secret and used to decrypt the data.

  In contrast, symmetric key cryptography uses the same secret key for both encryption and decryption.

- **Hash Functions**
  These are mathematical functions that take an input and return a fixed-size string (the hash). They are widely used to verify data integrity, as even a small change in the input leads to a significantly different output.

2.2 Advanced Blockchain Concepts

2.2.1 CAP Theorem

The CAP theorem is a fundamental principle in distributed systems, stating that it is impossible for a distributed system to simultaneously guarantee all three of the following properties:

- **Consistency (C)**: All nodes see the same data at the same time.
- **Availability (A)**: Every request receives a response, without guarantee that it contains the most recent write.
- **Partition Tolerance (P)**: The system continues to function despite arbitrary partitioning due to network failures.

In the context of blockchain, trade-offs must be made among these properties. For example, blockchains like Bitcoin favor availability and partition tolerance, potentially sacrificing strict consistency.

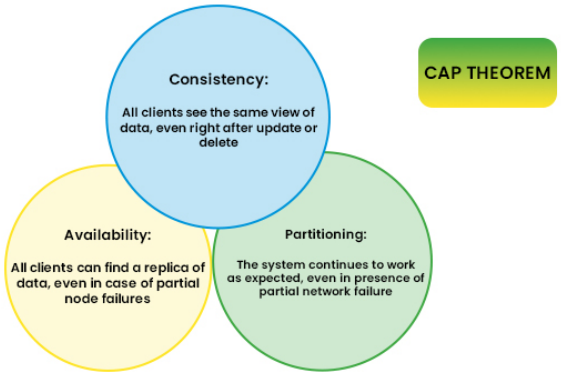

Figure 2.2: CAP Theorem

2.2.2 Byzantine Generals Problem and Consensus Mechanisms

Blockchain networks solve the **Byzantine Generals Problem**, which describes the difficulty of achieving consensus in distributed systems with potentially faulty or malicious actors.

Consensus mechanisms ensure that all participants agree on the state of the ledger. Key mechanisms include:

- **Proof of Work (PoW)**: Miners solve cryptographic puzzles; the first to solve can add a block.
- **Proof of Stake (PoS)**: Validators are chosen based on their stake in the network.

- **Delegated Proof of Stake (DPoS)**: Stakeholders vote to elect validators (witnesses).
- **Practical Byzantine Fault Tolerance (PBFT)**: Nodes reach consensus via majority voting.
- **Proof of Activity (PoA)**: A hybrid of PoW and PoS.
- **Proof of Authority**: Validators are chosen based on reputation rather than stake.

2.2.3 Cryptographic Primitives

Cryptographic primitives are the building blocks of blockchain security:

- **One-way Hash Functions**: Irreversible functions like SHA-256 provide data integrity and support the avalanche effect.
- **Symmetric Ciphers**: Encryption and decryption with the same key (e.g., AES, DES).
- **Asymmetric Ciphers**: Public/private key pairs (e.g., RSA, ECC) for secure communications.
- **Block and Stream Ciphers**: Encrypt data in fixed-size blocks or byte-by-byte streams.
- **Digital Signatures**: Validate authenticity and integrity using private/public key cryptography.
- **Message Authentication Codes (MACs)**: Verify message authenticity using a shared secret key.

2.2.4 Data Structures in Blockchain

Blockchain relies on several critical data structures:

- **Block**: Contains a header and transaction data.
- **Linked List**: Blocks are linked chronologically using cryptographic hashes.
- **Block Types**:
  - Main branch blocks
  - Side branch blocks
  - Orphan blocks
- **Merkle Tree**: Binary tree summarizing transaction hashes into a single Merkle root for efficient verification.

2.2.5 Block Structure and Metadata

Each block includes:

- **Header**: Contains block hash, timestamp, previous block hash, Merkle root, nonce, and difficulty target.
- **Transactions**: Verified and added to the block by network nodes.
- **Block Height**: Indicates the block's position in the chain (starting from 0 for the Genesis block).
- **Genesis Block**: The first block in the chain, serving as the anchor for all future blocks.

2.2.6 Linking of Blocks

Blocks are linked by including the hash of the previous block in the header of the new block. This ensures immutability and allows detection of tampering. If any block's contents change, its hash changes, invalidating subsequent blocks.

2.2.7 Merkle Tree: Efficient Verification

A Merkle Tree is a hierarchical structure for organizing and verifying data:

- Leaf nodes are hashes of individual transactions.
- Parent nodes are hashes of child node hashes.
- The **Merkle Root** is the top hash representing all transactions in the block.

Example (4 transactions: A, B, C, D):

- Hash A = $H(A)$, Hash B = $H(B)$, ...
- Hash AB = $H(H(A)||H(B))$, Hash CD = $H(H(C)||H(D))$
- Merkle Root = $H(HashAB||HashCD)$

Use Case: NFT Whitelisting with Merkle Trees

1. Hash participant identifiers (e.g., wallet addresses).
2. Build Merkle tree from hashes.
3. Publish Merkle root.
4. Participants can prove inclusion using a Merkle proof without revealing their full identifier.

This structure allows secure and privacy-preserving validation of eligibility for exclusive blockchain-based events.

CHAPTER 3

Elliptic Curve Cryptography

We start with a discussion of "Public-key Cryptography Systems", especially the **Diffie-Hellman** encryption.

3.1 Public-key Cryptography Systems

We will first define some basic definitions:-

Definition 1. *(Order of an element in a group) Let G be a finite abelian group, written multiplicatively. Let a be any element in G. The order of a is the order (number of elements) of the subgroup generated by a, denoted by $\langle a \rangle$ which consists of all powers of a. In other words, the order of a is the minimum value of i where $i > 0$ such that $a^i = 1$.*

Definition 2. *Let $\varphi(n)$ denote the Euler totient function which counts the number of integers from 1 to n (inclusive) which are coprime to n.*

Definition 3. *(Generator of finite field multiplicative group) An element $g \in \mathbb{F}_q^*$ is called a* generator *of the group $\mathbb{F}_q^*$, written multiplicatively, if for every $a \in \mathbb{F}_q^*$, we have $g^k = a$ for some integer k. In other words, the powers of g produce all elements in $\mathbb{F}_q^*$. If $q = p$ where p is a prime, a generator is called a* primitive root modulo p.

Theorem 1 (Euler Totient Function Property)**.** *Let n be a positive integer. Then the sum of Euler's totient function $\phi(d)$ over all divisors d of n equals n:*

$$\sum_{d|n} \phi(d) = n.$$

Proof. We proceed by strong induction on n.
Base Case: $(n = 1)$

- The only divisor is 1 itself
- $\phi(1) = 1$ since $\gcd(1, 1) = 1$
- Thus $\sum_{d|1} \phi(d) = 1 = n$

Inductive Step: Assume the theorem holds for all positive integers $k < n$. Consider two cases:
Case 1: n is prime $(n = p)$

- Divisors are $\{1, p\}$
- $\phi(1) = 1$ and $\phi(p) = p - 1$

- $\sum_{d|p} \phi(d) = 1 + (p-1) = p = n$

Case 2: n is composite

- Let p be the smallest prime dividing n
- Write $n = p^k m$ where $p \nmid m$ and $k \geq 1$
- The divisors are $d = p^i d'$ for $0 \leq i \leq k$ and $d' \mid m$

$$\begin{aligned}
\sum_{d|n} \phi(d) &= \sum_{i=0}^{k} \sum_{d'|m} \phi(p^i d') \\
&= \sum_{i=0}^{k} \sum_{d'|m} \phi(p^i)\phi(d') \quad \text{(since } \gcd(p^i, d') = 1\text{)} \\
&= \left(\sum_{i=0}^{k} \phi(p^i)\right)\left(\sum_{d'|m} \phi(d')\right) \\
&= p^k \cdot m \quad \text{(by induction hypothesis, as } m < n\text{)} \\
&= n
\end{aligned}$$

The last equality uses:

$$\sum_{i=0}^{k} \phi(p^i) = 1 + (p-1) + (p^2 - p) + \cdots + (p^k - p^{k-1}) = p^k$$

and by induction $\sum_{d'|m} \phi(d') = m$ since $m < n$.
Therefore, by strong induction, the theorem holds for all positive integers n. □

Theorem 2. *Every finite field $\mathbb{F}_q$ has a primitive element which is the generator of the multiplicatively written group of the field.*

Proof. To show that there exists a generator in $\mathbb{F}_q^*$, we need to prove the existence of an element of order $q-1$. Let a be an element of order r in the multiplicatively written group $\mathbb{F}_q^*$. Let $\langle a \rangle$ denote a cyclic group generated by a:

$$\langle a \rangle = \{1, a, a^2, \ldots, a^{r-1}\}$$

Every element in $\langle a \rangle$ is of the form a^k for some k. So $(a^k)^r = (a^r)^k = 1$ since $a^r = 1$. Therefore, a^k has order dividing r. It follows that a^k is a root of $X^r - 1$, a polynomial over $\mathbb{F}_q^*$. Then $\langle a \rangle$ forms the subset of roots of $X^r - 1$. Since the order of $\langle a \rangle$ is r and $X^r - 1$ has r roots, the number of elements in $\langle a \rangle$ equals the number of roots of $X^r - 1$. From the properties established before, the elements of $\langle a \rangle$ are the roots of $X^r - 1$. We know that a cyclic group of order n, $\mathbb{Z}/n\mathbb{Z}$ has $\varphi(n)$ generators where $\varphi(n)$ is the Euler totient function. It follows that the generators correspond to the integers which are coprime to n. Then $\langle a \rangle$ has $\varphi(r)$ generators or elements of order r.

Let $R = \{r_1, \ldots, r_m\}$ denote the set of the orders of the elements in $\mathbb{F}_q^*$. There are $\varphi(r_i)$ elements of order r_i for every i. Since $\mathbb{F}_q^*$ has order $q-1$, it follows from Lagrange's theorem that $r_i \mid (q-1)$ for all i. Then

$$q - 1 = |\mathbb{F}_q^*| = \sum_{r_i \in R} \varphi(r_i).$$

Let S be the set of all divisors of $q-1$. In other words, $S = \{r : r \mid (q-1)\}$. Then the Euler totient function satisfies the following property:

$$\sum_{r \in S} \varphi(r) = q - 1.$$

R is a subset of S because $r_i \mid (q-1)$ for all i. Then $S = R \cup (S \setminus R)$. We have that

$$\sum_{r \in S} \varphi(r) = \sum_{r \in R} \varphi(r) + \sum_{r \in S \setminus R} \varphi(r).$$

Since $\sum_{r \in S} \varphi(r) = \sum_{r \in R} \varphi(r) = q - 1$, it follows that $\sum_{r \in S \setminus R} \varphi(r) = 0$. Since $\varphi(r) > 0$ for all r, it follows that $S \setminus R$ is an empty set. Thus $S = R$. Since S is a set of elements dividing $q-1$, $q - 1 \in S$. It follows that $q - 1$ is also in R, the set of orders of elements in $\mathbb{F}_q^*$. Therefore, there is an element of order $q - 1$ in $\mathbb{F}_q^*$. Thus there exists at least one generator in $\mathbb{F}_q^*$. □

Discrete Logarithm Problem:- The mathematics of the method behind **Diffie-Hellman** is based on the discrete logarithm problem. If there is an element $z = x^y$, computing $y = log_x z$ is way more difficult in the finite field. Because of the use of the finite field than in the real field, this problem is called "discrete".

3.1.1 Diffie-Hellman Key Exchange

Protocol 1 (Diffie-Hellman Key Exchange)**.** *Let Alice and Bob establish a shared secret key over an insecure channel using the multiplicative group of a finite field. Consider implementation in* $(\mathbb{Z}/p\mathbb{Z})^*$ *where* g *is a generator.*

1. ***Public Parameters:*** *Alice and Bob agree on:*
   - *A large prime* p *(public modulus)*
   - *A generator* $g \in (\mathbb{Z}/p\mathbb{Z})^*$ *(public base)*
2. ***Alice's Private Computation:***
   - *Chooses private exponent* a *with* $1 < a < p - 1$
   - *Computes* $l \equiv g^a \pmod{p}$
   - *Sends* l *to Bob (public)*
3. ***Bob's Private Computation:***
   - *Chooses private exponent* b *with* $1 < b < p - 1$
   - *Computes* $m \equiv g^b \pmod{p}$
   - *Sends* m *to Alice (public)*
4. ***Shared Secret Derivation:***
   - *Alice computes* $s \equiv m^a \equiv (g^b)^a \pmod{p}$
   - *Bob computes* $s \equiv l^b \equiv (g^a)^b \pmod{p}$

The shared secret is $s \equiv g^{ab} \pmod{p}$.

Security Considerations.

- **Public Knowledge:** p, g, $l \equiv g^a \pmod{p}$, $m \equiv g^b \pmod{p}$
- **Private Knowledge:** a (Alice's secret), b (Bob's secret)

- **Eve's Challenge:** Given $g^a \pmod p$ and $g^b \pmod p$, compute $g^{ab} \pmod p$

The security relies on the **hardness of the Discrete Logarithm Problem (DLP)**:

- For large p, solving $g^x \equiv l \pmod p$ for x is computationally infeasible
- Using a generator g ensures maximal entropy in key space

The shared key s is typically used for symmetric encryption, while the asymmetric scheme (like ElGamal) may be used to encrypt the symmetric key itself. □

3.2 Elliptic Curve Cryptography

Researchers spent quite a lot of time trying to explore cryptographic systems based on more reliable trapdoor functions (going one way is easy but coming back is computationally very hard) and in 1985 succeeded by discovering a new method, namely the one based on elliptic curves which were proposed to be the basis of the group for the discrete logarithm problem. Researchers believe that elliptic curves guarantee more security and provide with much smaller key sizes than other groups. For better understanding of the extent, let us use the visualization that compares the amount of energy one needs to break a cryptographic system with how much water that energy could boil. For example, a 228-bit RSA key can be broken using less energy than that required to boil a teaspoon of water. However, one can equate the amount of energy needed to break a 228-bit elliptic curve key with the energy used to boil all water on earth.

Definition 4 (Elliptic Curve)**.** *An* elliptic curve *E over a field K is a smooth cubic curve consisting of:*

- *All points $(x, y) \in K \times K$ satisfying the Weierstrass equation:*

$$y^2 = x^3 + ax + b$$

- *A distinguished point $\mathcal{O}$ called the "point at infinity"*

Remarks:

1. *This is the simplified Weierstrass form, valid when* $\mathrm{char}(K) \neq 2, 3$
2. *The curve is non-singular when the discriminant* $\Delta = -16(4a^3 + 27b^2) \neq 0$
3. *Non-singularity ensures the cubic $x^3 + ax + b$ has distinct roots in $\overline{K}$*

3.2.1 Group Law on Elliptic Curves

The points of E form an abelian group under the chord-and-tangent rule:

- **Identity:** $\mathcal{O}$ is the identity element.
- **Inverse:** The inverse of $P = (x, y)$ is $-P = (x, -y)$.
- **Addition:** For $P = (x_1, y_1)$, $Q = (x_2, y_2)$:
  1. If $P = \mathcal{O}$, then $P + Q = Q$.
  2. If $Q = -P$, then $P + Q = \mathcal{O}$.
  3. If $P \neq Q$ and $x_1 \neq x_2$, the line PQ intersects E at a third point $R = (x_3, y_3)$, and $P + Q = -R = (x_3, -y_3)$:

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}, \quad x_3 = \lambda^2 - x_1 - x_2, \quad y_3 = \lambda(x_1 - x_3) - y_1$$

  4. If $P = Q$ (point doubling), the tangent at P intersects E at R, and $2P = -R$:

$$\lambda = \frac{3x_1^2 + a}{2y_1}, \quad x_3 = \lambda^2 - 2x_1, \quad y_3 = \lambda(x_1 - x_3) - y_1$$

3.2.2 Elliptic Curve Discrete Logarithm Problem (ECDLP)

Given points $P, Q \in E(K)$ where $Q = kP$ for some integer k, the ECDLP is to find k. The hardness of ECDLP underpins ECC security. Known attacks include:

- **Baby-step giant-step:** Time/memory trade-off, $O(\sqrt{n})$ complexity.
- **Pollard's rho:** Probabilistic, $O(\sqrt{n})$ complexity, parallelizable.
- **Index calculus:** Ineffective for well-chosen elliptic curves.

For a curve with subgroup order n (prime), the best attacks require $\approx \sqrt{n}$ operations.

3.2.3 Security Advantages

ECC achieves equivalent security with smaller keys than other systems:

RSA/DSA Key Size	ECC Key Size	Security Level
1024 bits	160 bits	80 bits
2048 bits	224 bits	112 bits
3072 bits	256 bits	128 bits
15360 bits	512 bits	256 bits

Reasons for efficiency:

- Absence of subexponential-time attacks on general elliptic curves.
- Larger search space per bit due to group structure.
- Smaller keys reduce storage, bandwidth, and computation overhead.

3.2.4 Cryptographic Setup

To use ECC in practice:

1. **Parameter selection:**
   - Choose a prime p (for $\mathbb{F}_p$) or binary field $\mathbb{F}_{2^m}$.
   - Select coefficients a, b defining $E : y^2 = x^3 + ax + b$ over the field.
   - Verify $\Delta \neq 0$ and compute the group order $N = \#E$ using Schoof's algorithm.
   - Choose a base point G with prime order n satisfying $n \mid N$ and $n > 2^{160}$.
2. **Key generation:**
   - Private key: Random integer $d \in [1, n-1]$.
   - Public key: Point $Q = dG$.
3. **Encryption/decryption:** Use protocols like ECIES or ECDH for key exchange.

3.2.5 Example: Curve over $\mathbb{F}_{23}$

Let $E : y^2 = x^3 + x + 1$ over $\mathbb{F}_{23}$ ($\Delta = -16(4 + 27) \equiv 8 \not\equiv 0 \pmod{23}$). Points include:

$$(0, 1), (0, 22), (1, 7), (1, 16), (3, 10), (3, 13), (4, 0), \ldots, \mathcal{O}$$

Total $\#E = 28$. With $G = (0, 1)$ of order 14:

$$2G = (0, 1) + (0, 1) = (1, 7), \quad 3G = (0, 1) + (1, 7) = (3, 10), \quad \ldots$$

Given $Q = (3, 10)$, solving $Q = kG$ yields $k = 3$.

CHAPTER 4

Bitcoin: A Peer-to-Peer Electronic Cash System

This chapter will be a report on the study of the paper on Bitcoin by ”Satoshi Nakamoto”.

4.1 Introduction

The paper *“Bitcoin: A Peer-to-Peer Electronic Cash System”* by Satoshi Nakamoto, published in 2008, proposes a decentralized digital currency system, referred to as Bitcoin. The main motivation behind the proposal is to eliminate the need for trusted third parties (such as banks or payment processors) in online transactions.

4.2 Motivation

Traditional electronic payment systems rely heavily on centralized financial institutions, which increases transaction costs, imposes settlement delays, and limits privacy. Moreover, such systems are vulnerable to fraud due to the reversible nature of transactions. Nakamoto argues that a peer-to-peer system based on cryptographic proof can solve these issues, enabling direct transfers between parties without the need for trust.

4.3 System Overview

Bitcoin uses a decentralized network of nodes to maintain a public ledger known as the *blockchain*. Transactions are broadcasted to the network and grouped into blocks. These blocks are then linked in a chronological and cryptographically secure chain.

4.3.1 Proof-of-Work

To add a block to the blockchain, a node (miner) must solve a computationally difficult puzzle called the *proof-of-work* (PoW). This mechanism prevents double-spending and ensures that adding new blocks requires a significant amount of computational effort. The PoW is based on the SHA-256 hash function, and the difficulty is adjusted periodically to maintain a constant average block generation time.

4.3.2 Incentive Mechanism

Miners are rewarded with newly minted bitcoins and transaction fees for performing PoW and validating transactions. This provides a financial incentive to contribute computing power to

the network and ensures the security and integrity of the system.

4.4 Consensus and Security

Consensus in the Bitcoin network is achieved by following the longest chain rule—nodes always consider the longest valid chain of blocks as the correct one. This discourages tampering, as altering a past block would require redoing the PoW for all subsequent blocks, which becomes computationally infeasible over time.

4.5 Privacy

While all transactions are publicly visible on the blockchain, user identities are pseudonymous. Bitcoin addresses are not directly linked to users' real-world identities. However, Nakamoto acknowledges that complete privacy is not guaranteed and depends on the use of new addresses for each transaction.

4.6 Conclusion

The paper introduces a revolutionary model for decentralized digital currency, resolving the double-spending problem without relying on centralized trust. It combines cryptographic techniques, peer-to-peer networking, and economic incentives to create a robust and self-sustaining system. Bitcoin laid the foundation for the development of modern cryptocurrencies and blockchain technology.